Microcontroller Technical Information

32-Bit Microcontroller V850E/ME2 Usage Restrictions	Document No.	ZBG-CC-07-0022	1/1
	Date issued	October 1, 2007	
	Issued by	2nd Product Solution Group Multipurpose Microcomputer Systems Division Microcomputer Operations Unit NEC Electronics Corporation	
Related documents V850E/ME2 Hardware User's Manual: U16031EJ5V0UD00 (5th edition) V850E1 Architecture User's Manual: U14559EJ3V0UM00 (3rd edition)	Notification classification	√	Usage restriction
			Upgrade
			Document modification
			Other notification

1. Affected products

   V850E/ME2
   - μPD703111
   - μPD703111A
   - μPD703111B (added in this edition)

2. Notification

   The previous notification (document number: ZBG-CC-04-0018) stated that restriction No. 11 (restriction on A/D converter) will be corrected in the μPD703111AGM of rank E, but the part number has been changed. Restriction No. 11 has been corrected and released as version B (part number: μPD703111BGM).

   Note: The μPD703111AGM of rank E does not exist.

3. List of restrictions

   The restriction history and detailed information is described in the attachment.

4. Document revision history

32-Bit Microcontroller V850E/ME2 - Usage Restrictions

Document Number	Date Issued	Description
SBG-DT-03-0154-E (1st edition)	May 23, 2003	Newly created.
SBG-DT-03-0220 (2nd edition)	August 19, 2003	Addition of new bugs (No. 3 and No. 4)
SBG-DT-04-0060 (3rd edition)	February 9, 2004	Addition of new bugs (No. 5 to No. 8)
SBG-DT-04-0101 (4th edition)	March 2, 2004	Addition of new bugs (No. 9 and No. 10)
ZBG-CC-04-0018 (5th edition)	September 22, 2004	Addition of new bug (No. 11)
ZBG-CC-07-0022 (latest edition)	October 1, 2007	Revision of document

List of Usage Restrictions in V850E/ME2

1. Product Version

μPD703111: Rank K, E

μPD703111A: Rank K

μPD703111B: Rank K (product in which restriction No. 11 is corrected)

* The rank is indicated by the letter appearing as the 5th digit from the left in the lot number marked on each product.

2. Product History

• μPD703111

No.	Bugs	Rank	
		K	E
1	Restriction on successive access to SDRAM and SRAM interface device	Δ	○
2	Bug in bus hold function	Δ	○
3	Bug in flyby DMA transfer from SDRAM to external I/O	Δ	Δ
4	Bug in USB function	Δ	Δ
5	Restriction on conflict between sld instruction and interrupt	Δ	Δ
6	Restriction on 2-cycle DMA transfer when using speculative read function	Δ	Δ
7	Restriction on speculative read function used for SDRAM with 32-bit bus	Δ	Δ
8	Restriction on speculative read function when undivided external bus is used	Δ	Δ
9	Restriction on 2-cycle DMA transfer to internal data RAM	Δ	Δ
10	Restriction on speculative read function	Δ	Δ
11	Restriction on A/D converter	Δ	Δ

○: Bug does not occur, Δ: Bug will also apply in future

• μPD703111A

No.	Bugs	Rank
		K
1	Restriction on successive access to SDRAM and SRAM interface device	○
2	Bug in bus hold function	○
3	Bug in flyby DMA transfer from SDRAM to external I/O	Δ
4	Bug in USB function	○
5	Restriction on conflict between sld instruction and interrupt	Δ
6	Restriction on 2-cycle DMA transfer when using speculative read function	Δ
7	Restriction on speculative read function used for SDRAM with 32-bit bus	Δ
8	Restriction on speculative read function when undivided external bus is used	Δ
9	Restriction on 2-cycle DMA transfer to internal data RAM	Δ
10	Restriction on speculative read function	Δ
11	Restriction on A/D converter	Δ

○: Bug does not occur, Δ: Bug will also apply in future

• μPD703111B

No.	Bugs	Rank
		K
1	Restriction on successive access to SDRAM and SRAM interface device	○
2	Bug in bus hold function	○
3	Bug in flyby DMA transfer from SDRAM to external I/O	Δ
4	Bug in USB function	○
5	Restriction on conflict between sld instruction and interrupt	Δ
6	Restriction on 2-cycle DMA transfer when using speculative read function	Δ
7	Restriction on speculative read function used for SDRAM with 32-bit bus	Δ
8	Restriction on speculative read function when undivided external bus is used	Δ
9	Restriction on 2-cycle DMA transfer to internal data RAM	Δ
10	Restriction on speculative read function	Δ
11	Restriction on A/D converter	○

○: Bug does not occur, Δ: Bug will also apply in future

3. Details of Usage Restrictions

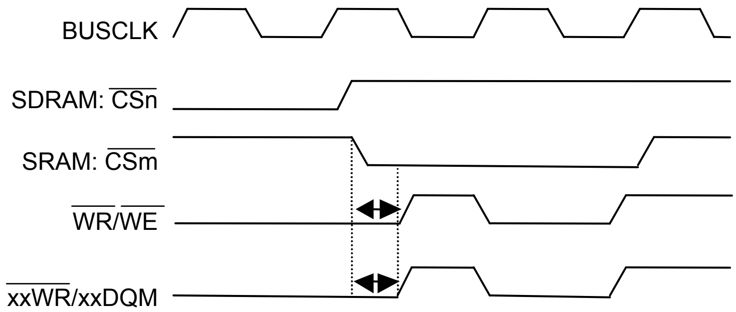
No. 1 Restriction on successive access to SDRAM and SRAM interface device

[Description]

When an access to the SRAM interface device occurs immediately after an access to the SDRAM occurs, the data written to the SRAM interface device may be incorrect.

The UU/UL/LU/LLWR signals (hereinafter referred to as the xxWR signal) and UU/UL/LU/LLDQM signals (hereafter referred to as the xxDQM signal) are alternate-function pins (xxWR/xxDQM), and the WR signal and WE signal are alternate-function pins (WR/WE).

When an access to the SRAM interface device occurs immediately after an access to the SDRAM occurs, the rise (inactive timing) of the xxDQM (xxWR) signal or WE (WR) signal overlaps the SRAM interface device cycle, which may cause erroneous write (see the figure below).



Erroneous write may occur because the $\overline{\text{CSn}}$ signal of the SDRAM falls before the $\overline{\text{xxWR}}$/xxDQM and $\overline{\text{WR}}$/$\overline{\text{WE}}$ signals rise.

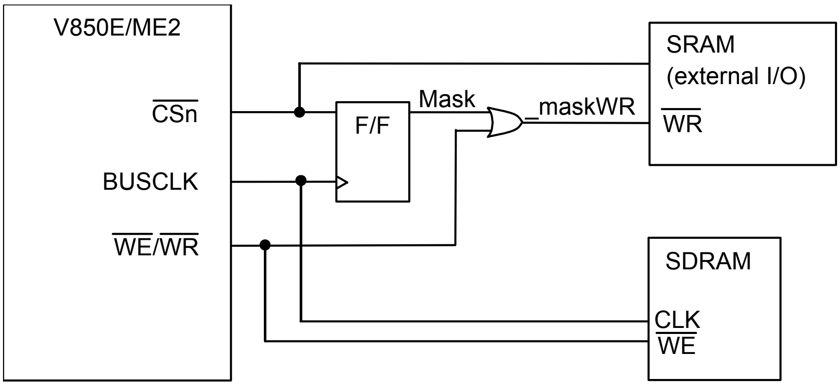
[Workaround]

Implement any of the following workarounds.

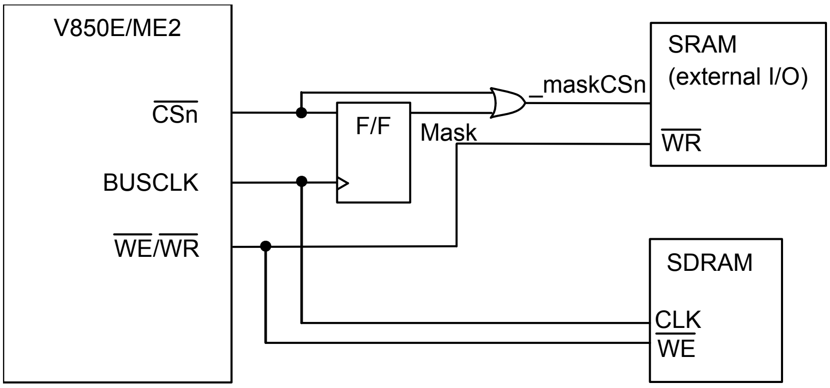
- Use the IOWR signal, instead of the xxWR or WR signal, to access the SRAM interface device. In addition, configure a circuit so that the xxBE and IOWR signals are logically ORed externally if byte control is required.

  In this workaround, however, <u>flyby DMA transfer between the memory and external I/O cannot be performed when the transfer target on the memory side is other than SDRAM</u> (e.g. flyby DMA transfer between the memory and SRAM). In addition, the IOWR signal must be activated by setting the IOEN bit of the BCP register to 1 after program transfer to the internal instruction RAM is complete.

- When the IOWR signal cannot be used, sample the CS signal to the SRAM interface device using BUSCLK as shown in the figure below, and mask the write signal. Note, however, that the WR signal input (fall) to the SRAM interface device may be delayed, depending on conditions such as the BUSCLK frequency. In such a case, insert a wait as necessary.



- Sample the CS signal to the SRAM interface device using BUSCLK as shown in the figure below, and mask the CS signal. In this case, insert an additional wait in the address setup wait cycle for the SRAM interface device (using the ASC register).



- When using an ASIC, design synchronization so that the CSn, xxWR, and WR signals are sampled using BUSCLK.
- This bug can be avoided by inserting an idle state using the BCC register if it is clear that the SRAM interface device cycle only occurs immediately after an SDRAM read cycle.

[Permanent workaround]

The following function has been added to bits 0, 4, 6, and 7 of the PFCCS register in products of rank E or later.

Port CS function control register (PFCCS)

	7	6	5	4	3	2	1	0	Address	Initial value
PFCCS	**CSDC7**	**CSDC6**	PFCCS5	**CSDC4**	0	PFCCS2	0	**CSDC0**	FFFFF049H	00H

Bit position	Bit name	Function
0, 4, 6, 7	CSDCn	When this bit is set (1), the timing at which the corresponding chip select signal (CSn) falls is delayed by one clock. The output timing of signals other than CSn is not affected.

Remark n = 0, 4, 6, 7

Cautions 1. Be sure to set the BTn1 and BTn0 bits of the BCT0 and BCT1 registers to 00 or 01 so that the device to which this restriction applies (erroneous write may occur) is connected to the CS space to which the CSDCn bit is set (1).

2. Do not change the value of the CSDCn bit for the CS space where the program under execution is allocated.

3. Be sure to insert one or more address setup waits (required number of waits + 1) in the CS space for which the CSDCn bit is set (1) using the ASC register.

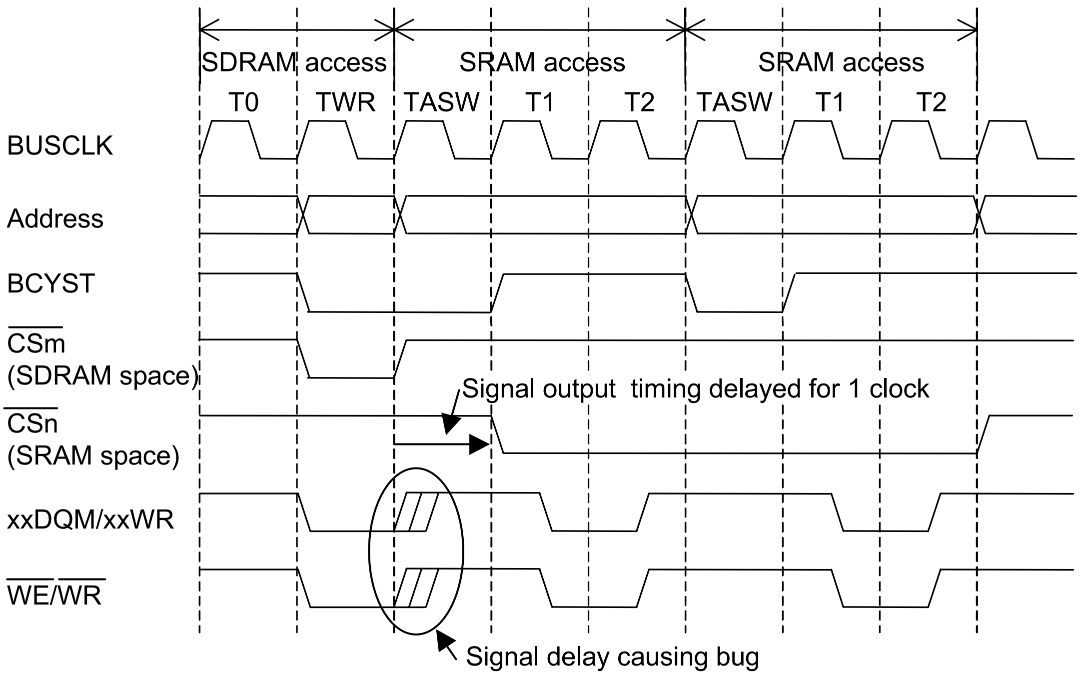
4. Be sure to set bits 1 and 3 to 0.

- SRAM access immediately after SDRAM access

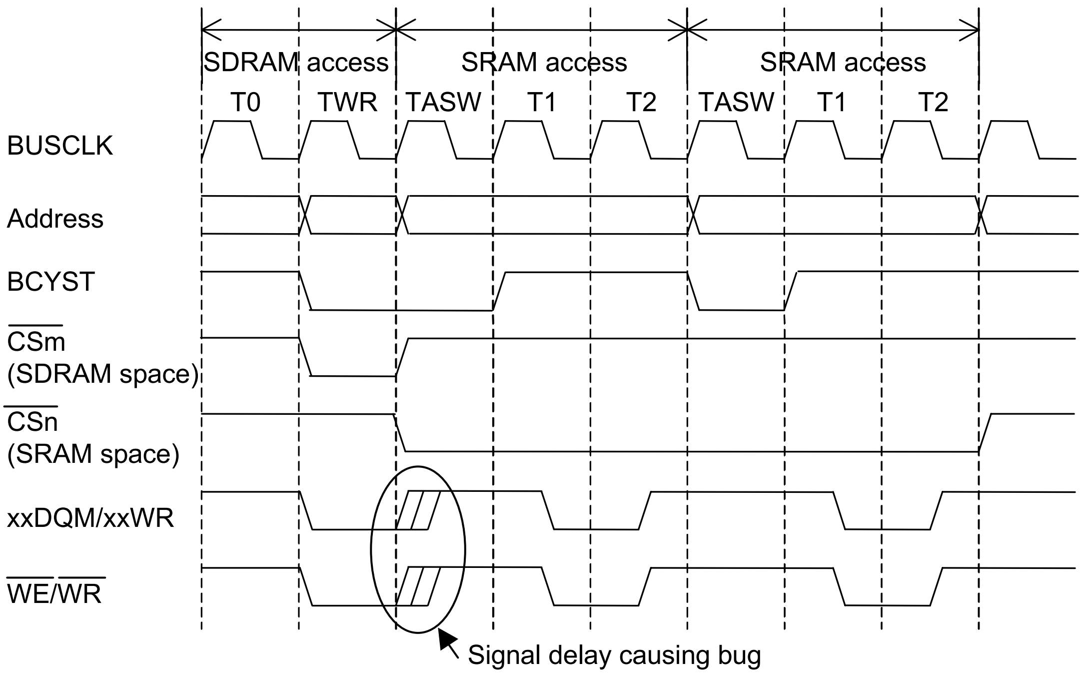
  (SRAM access is performed twice in succession immediately after SDRAM access)

  SRAM settings: Address setup waits = 1, data waits = 0, CSCDn = 1

1. CSDCn = 1 (When a CS signal in SRAM cycle is delayed for 1 clock)



2. CSDCn = 0 (When no additional function is used)



Remark These are the timing diagram when no T0 state is inserted (this bug does not occur when a T0 state is inserted). See the user's manual for the detailed timing at which a T0 state is inserted.

No. 2 Bug in bus hold function

[Description]

(1) External bus deadlock

The external bus may deadlock when BUSCLK is used at 1/3 or 1/4 of the internal operating frequency, when one of the following operations and an active level input (fall) to the HLDRQ pin conflict, and when a speculative read cycle to an external memory access occurs.

- Releasing software STOP mode or IDLE mode
- SCRn register write cycle to SDRAM (n = 1, 3, 4, or 6)
- Refresh cycle to SDRAM (including self-refresh cycle by the SELFREF pin)

[Conditions under which this bug occurs]

This bug may occur when all the following conditions are satisfied.

- The bus hold function is used.
- The speculative read function for the external memory access is active.
- BUSCLK is used at 1/3 or 1/4 of the internal operating frequency.
- SDRAM is used, or software STOP mode or IDLE mode is used.

(2) Illegal output of HLDAK signal

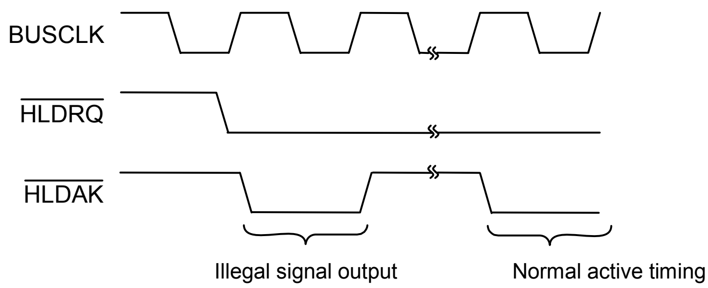
When one of the following operations and an active level input (fall) to the HLDRQ pin conflict, the HLDAK signal may become active for 1 BUSCLK illegally, then inactive, and active again after several BUSCLKs have elapsed (the second active operation is the normal HLDAK signal operation) (see the figure below).

- Releasing software STOP mode or IDLE mode
- SCRn register write cycle to SDRAM (n = 1, 3, 4, or 6)
- Refresh cycle to SDRAM (including self-refresh cycle by the SELFREF pin)

[Conditions under which this bug occurs]

This bug may occur when both the following conditions are satisfied.

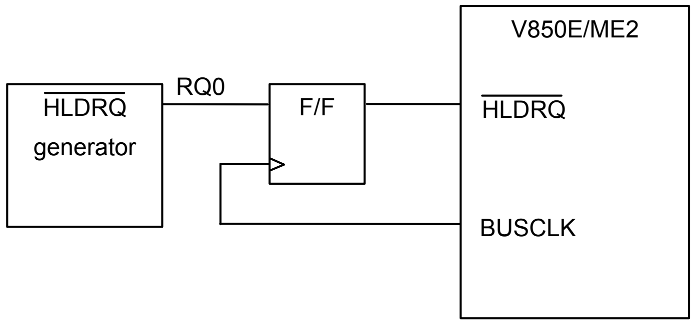
- The bus hold function is used.
- SDRAM is used, or software STOP mode or IDLE mode is used.



[Workaround]

Insert an external circuit to avoid <1> and <2> above.

Insert an external circuit so that the HLDRQ signal does not change in the setup time and hold time periods of the HLDRQ signal. The circuit example is shown below.



No. 3 Bug in flyby DMA transfer from SDRAM to external I/O

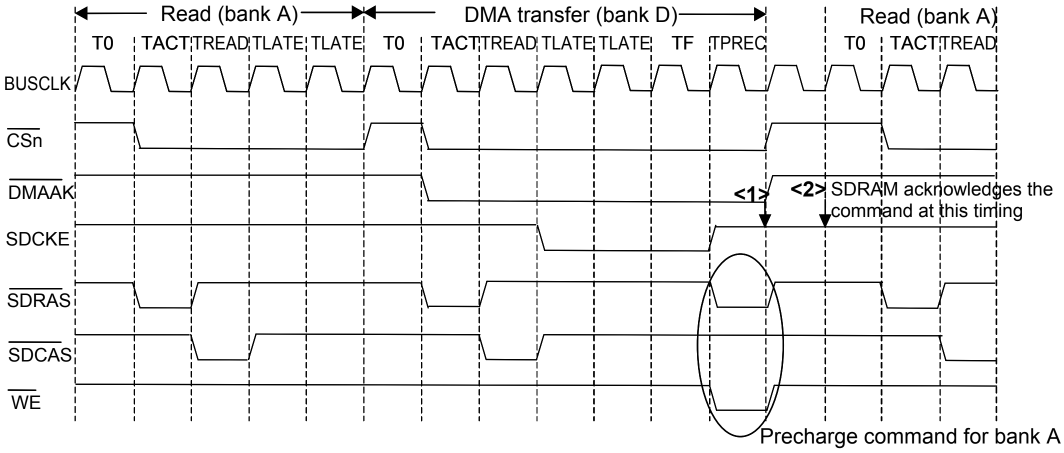
[Description]

If a bank change occurs in the SDRAM when a read or write operation to the SDRAM and a DMA flyby transfer to the SDRAM are performed successively, a precharge command is generated at an illegal timing. Consequently, the subsequent SDRAM access may be illegal. In addition, if the latency with respect to the SDRAM is 3 or if the data size of the DMA transfer is set to a value larger than the bus width of the SDRAM, the SDRAM access command after the flyby transfer may be generated at an illegal timing, which may cause the subsequent SDRAM access to be illegal, regardless of bank change occurrence.

This bug occurs only when performing flyby DMA transfer from the SDRAM to external I/O. This bug does not occur when performing flyby DMA transfer from external I/O to the SDRAM or when performing 2-cycle transfer.

[Details]

The V850E/ME2 controls waits when performing flyby DMA transfer to the SDRAM by inactivating the SDCKE signal (L level). When the SDRAM has accessed a bank (hereafter referred to as A) and this is followed by a DMA transfer to another bank (hereafter referred to as D), the V850E/ME2 activates the SDCKE signal (H level) and subsequently issues a precharge command (<1> in the figure below). However, the SDRAM can acknowledge commands only at the fall of the clock immediately after the H level of the SDCKE signal is detected (<2> in the figure below). Consequently, the precharge command output from the V850E/ME2 is not acknowledged, and the SDRAM access to bank A performed after the DMA transfer becomes illegal.



[Workaround]

Use 2-cycle transfer instead of flyby transfer when performing DMA transfer from the SDRAM to external I/O.

No. 4 Bug in USB function

[Description]

(1) When the SNDSTL bit of the UF0SDS register is set in a specific period (4 USB clocks) during an IN transfer token phase for EP0, the data phase continues to transmit "00H" and cannot be completed normally (see the figure below for the detailed timing).

(2) When the SNDSTL bit of the UF0SDS register is set in a specific period (4 USB clocks) during an OUT transfer token phase for EP0, the device returns ACK to the host in the data phase even if the buffer is in the NAK status.

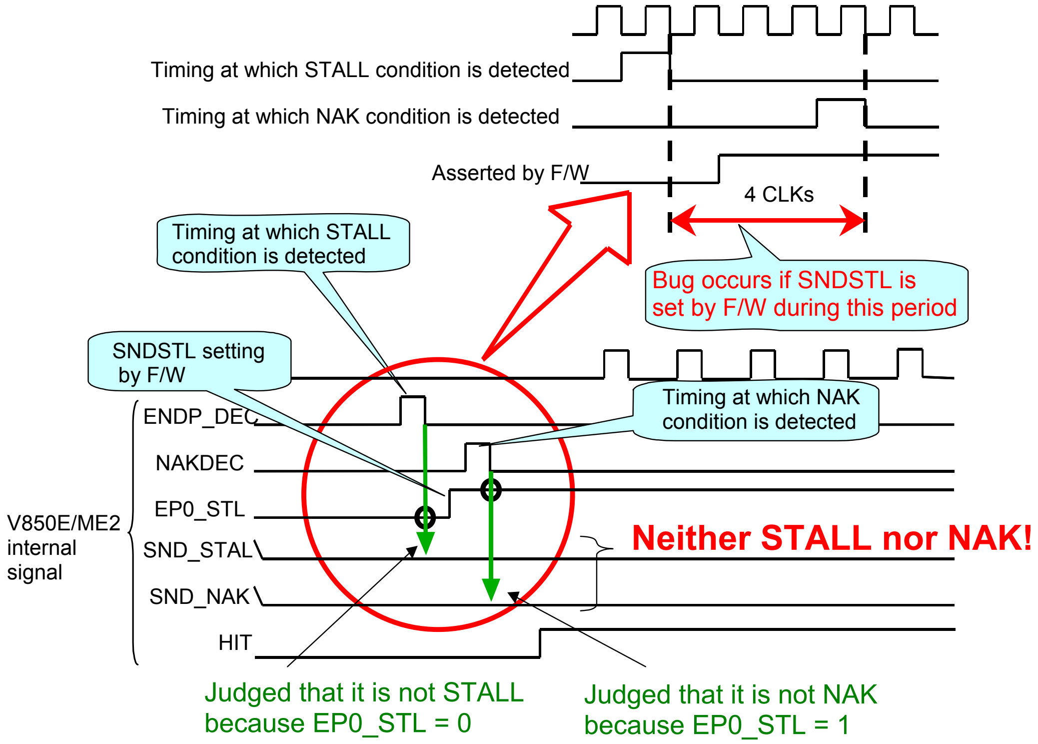
Remarks **1.** In the case of (2) above, the OUT transfer to the V850E/ME2 seems to complete normally as viewed from the host. STALL acknowledgment set by software is performed on the token after the OUT transfer, so the status is restored normally without implementing any measures.

2. This bug does not occur when using endpoints other than EP0 because setting the SNDSTL bit by software during a transaction is made invalid by hardware.

3. This bug does not occur when using an automatic response request because setting the SNDSTL bit by software is made invalid by hardware.

- Detailed timing at which this bug occurs

When using the USB function of the V850E/ME2, there is a 4-USB-clock difference between the timing at which the STALL acknowledgment ready status is detected in the device and when the next NAK response is detected. If the SNDSTL bit of the UF0SDS register is set by software during this 4-USB-clock period, neither STALL nor NAK is recognized, which causes this bug.



[Workaround]

Design the host driver so that a device request that is not recognized by the function driver is not output. Consult an NEC Electronics sales representative or distributor if implementation of this workaround is difficult.

No. 5 Restriction on conflict between **sld** instruction and interrupt

[Description]

If a conflict occurs between the decode operation of the instruction (<2> in the examples) immediately before the **sld** instruction (<3> in the examples) following a special instruction (<1> in the examples) and an interrupt request before execution of the special instruction is complete, the execution result of the special instruction may not be stored in a register.

This bug may only occur when the same register is used as the destination register of the special instruction and the **sld** instruction, and when the register value is referenced by the instruction followed by the **sld** instruction.

Special instruction:

- **ld** instruction: ld.b, ld.h, ld.w, ld.bu, ld.hu
- **sld** instruction: sld.b, sld.h, sld.w, sld.bu, sld.hu
- Multiply instruction: mul, mulh, mulhi, mulu

Examples of instruction sequence that may cause the bug:

Example 1:

<1> ld.w [r11],**r10** : <2> mov **r10** ,r28 <3> sld.w 0x28 ,r10	This bug occurs when the decode operation of **mov** (<2>) immediately before **sld** (<3>) and interrupt request servicing conflict before execution of the special instruction **ld** (<1>) is complete.

Example 2:

(1) ld.w [r11],**r10** : <2> cmp imm5, **r10** <3> sld.w 0x28 ,r10 <4> bz label	This bug occurs when the decode operation of **comp** (<2>) immediately before **sld** (<3>) and interrupt request servicing conflict before execution of the special instruction **ld** (<1>) is complete. As a result, the compare result of **comp** becomes illegal, which may cause illegal operation of the branch instruction **bz** (<4>).

Example 3:

<1> ld.w [r11],**r10** : <2> add imm5, **r10** <3> sld.w 0x28 ,**r10** <4> setf r16	This bug occurs when the decode operation of **add** (<2>) immediately before **sld** (<3>) and interrupt request servicing conflict before execution of the special instruction **ld** (<1>) is complete. As a result, the result of **add** and the flag become illegal, which may cause illegal operation of the **setf** (<4>).

[Conditions under which this bug occurs]

This bug may occur when all the following conditions (1) to (3) are satisfied.

(1) Either condition (I) or (II) is satisfied

Condition (I):

The same register is used as the destination register of a special instruction (see below) and the subsequent **sld** instruction and as the source register (reg1) of an instruction shown below followed by the **sld** instruction. (See Example 1.)

mov **reg1**,reg2	not **reg1**,reg2	satsubr **reg1**,reg2	satsub **reg1**,reg2
satadd **reg1**,reg2	or **reg1**,reg2	xor **reg1**,reg2	and **reg1**,reg2
tst **reg1**,reg2	subr **reg1**,reg2	sub **reg1**,reg2	add **reg1**,reg2
cmp **reg1**,reg2	mulh **reg1**,reg2		

Condition (II):

The same register is used as the destination register of a special instruction (see below) and the subsequent **sld** instruction and as the source register (reg2) of an instruction shown below followed by the **sld** instruction. (See Examples 2 and 3.)

not reg1,**reg2**	satsubr reg1,**reg2**	satsub reg1,**reg2**	satadd reg1,**reg2**
satadd imm5,**reg2**	or reg1,**reg2**	xor reg1,**reg2**	and reg1,**reg2**
tst reg1,**reg2**	subr reg1,**reg2**	sub reg1,**reg2**	add reg1,**reg2**
add imm5,**reg2**	cmp reg1,**reg2**	cmp imm5,**reg2**	shr imm5,**reg2**
sar imm5,**reg2**	shl imm5, **reg2**		

Special instruction:

- **ld** instruction: ld.b, ld.h, ld.w, ld.bu, ld.hu
- **sld** instruction: sld.b, sld.h, sld.w, sld.bu, sld.hu
- Multiply instruction: mul, mulh, mulhi, mulu

(2) When the execution result of the special instruction (see above) has not been stored in the destination register before execution of the instruction (instruction of condition (I) or (II)) immediately before the **sld** instruction starts in the CPU pipeline.

(3) When the decode operation of the instruction (instruction of condition (I) or (II)) immediately before the **sld** instruction and interrupt request servicing conflict.

[Workaround]

- Workaround using assembler

  Implement any of the following workarounds.

  - Insert a **nop** instruction between instruction <2> in the above instruction sequence and the **sld** instruction (<3> in the examples) immediately after <2>.
  - Do not use the same register as is used by instruction <2> in the above instruction sequence for the **sld** instruction (<3> in the examples) immediately after <2>.

- Workaround using compiler

  Regard this item as a usage restriction on the CPU function. A compiler that can automatically suppress generation of the instruction sequence that may cause the bug will be provided. Provision of the compiler varies depending on the compiler currently being used. Consult an NEC Electronics sales representative or distributor if you are using a compiler other than one below.

  - **When NEC Electronics compiler CA850 is used**

    CA850 V2.61, which includes the countermeasure function for this bug, has been released via the online delivery service (ODS; user registration is required via mail after purchasing the compiler) on the following website.

    URL: http://www.necel.com/micro/ods/eng/index.html

  - **When GHS compiler CC850 is used**

    The distributor (Advanced Data Controls Corp.) upgraded Multi 4.0 (Rel. 7.0.0) and Multi 3.5.1 (Rel. 6.5.3) so as to include the countermeasure function for this bug, so contact the distributor if using these versions. When using versions other than above, separately consult the distributor.

    Inquiry:

    TEL: +81-3-3576-6805

    E-mail: upgv850e@adac.co.jp

No. 6 Restriction on 2-cycle DMA transfer when using speculative read function

[Description]

When performing 2-cycle DMA transfer in which the use of the speculative read function is selected for the CS space at the DMA transfer destination (side to be written), the TC signal may become active twice (normally it becomes active once) and the DMA transfer end interrupt occurs twice (normally it occurs once) when DMA transfer is complete. However, the DMA transfer itself is completed normally.

This bug does not occur during flyby transfer.

Conditions under which this bug occurs:

This bug may occur when all the following conditions are satisfied.

- The use of the speculative read function is selected for the CS space in the DMA transfer destination (side to be written).
- The address on which a speculative read is performed and the line address to which the last DMA transfer data is written are the same line address (the same line address means that A25 to A4 are the same in an same CS space).
- A divided internal system clock is used for the external BUSCLK.

[Workaround]

Regard this item as a restriction on using the DMA function.

Implement any of the following workarounds to avoid this bug.

- Do not select the use of the speculative read function for the CS space in the DMA transfer destination (side to be written).
- Set the DMA transfer destination address so that the address on which a speculative read is performed and the line address to which the last DMA transfer data is written are not the same line address.

No. 7 Restriction on speculative read function used for SDRAM with 32-bit bus

[Description]

If all the conditions shown below are satisfied when the use of the speculative read function is selected for SDRAM with a 32-bit bus, the external bus may deadlock immediately after the single-write operation described in the condition, or the single-write operation may be executed twice.

Conditions under which this bug occurs:

- A read request is generated for the same line address (A25 to A4 are the same in the same CS space) as the speculative read start address inside the CPU during a speculative read cycle.
- A single-write request is generated inside the CPU immediately before reading the item hit by the speculative read is complete.

[Workaround]

Regard this item as a restriction on using the speculative read function.

Do not use the speculative read function for the SDRAM with 32-bit bus.

No. 8 Restriction on speculative read function when undivided external bus is used

[Description]

When an undivided internal system clock is used for the external BUSCLK and a sequential read access (twice or four-time consecutive read operation) that hits a speculative read buffer inside the CPU is performed during a speculative read cycle, the read cycle inside the CPU may deadlock. (The speculative read cycle for the external memory is completed but no bus cycles can subsequently be generated.)

[Workaround]

Regard this item as a restriction on using the speculative read function.

Implement any of the following workarounds to avoid this bug.

- Do not use an undivided internal system clock for the external BUSCLK.
- Do not use the speculative read function.

No. 9 Restriction on 2-cycle DMA transfer to internal data RAM

[Description]

When performing any of the following 2-cycle DMA transfers, the TC signal output may become active twice (normally it becomes active once) and the DMA transfer end interrupt may occur twice (normally it occurs once) when the last data has been transferred.

- In the case of 2-cycle DMA transfer from external memory to internal data RAM
  * This restriction applies only when the speculative read function (set by the LBC0 or LBC1 register) is used for the CS space in the DMA transfer source (side to be read).
- In the case of 2-cycle DMA transfer from on-chip peripheral I/O to internal data RAM
  * This restriction applies regardless of the speculative read setting.

[Workaround]

Implement any of the following workarounds to avoid this bug.

- In the case of 2-cycle DMA transfer from external memory to internal data RAM
  Do not make the speculative read function valid for the CS space in the DMA transfer source (side to be read).

- In the case of 2-cycle DMA transfer from on-chip peripheral I/O to internal data RAM
  Do not use the TC signal. In addition, execute the processing of (1) and (2) below consecutively for the excessive DMA transfer end interrupt in the DMA transfer end interrupt servicing routine. After the processing of (2) is executed, by executing the application processing that should be performed in the normal operation and restoring from the interrupt, occurrence of the second DMA transfer end interrupt can be suppressed.

  (1) Set the write access synchronization control register (WAS) to 00H.
  (2) Clear bit 7 (DMAIFn) of the interrupt control register (DMAICn) of the same channel as the DMA transfer end interrupt currently being serviced to 0.

Write access synchronization control register (WAS)

When an external device is written, even if the write operation by the CPU using the write buffer function is complete, writing to the external device may not be complete. The WAS register is used to complete writing all data in the write buffer to the external device.

This register is write-only, in 8-bit units.

	7	6	5	4	3	2	1	0	Address	Initial value
WAS	0	0	0	0	0	0	0	0	FFFFF49CH	Undefined

Caution Do not write a value other than 00H; otherwise the operation is not guaranteed.

* This register is incorporated in V850E/ME2 devices that have already been shipped but its information was not disclosed to the user. The descriptions of this register and its specifications have been added to the user's manual.

No. 10 Restriction on speculative read function

[Description]

If a CS space that is connected to SRAM for which the speculative read function is not valid or to a page ROM is read-accessed immediately after execution of a speculative read cycle for a CS space for which the speculative read function is valid (using the LBC0 or LBC1 register), the CPU may fail to load data for the CS space for which the speculative read function is not valid at the correct timing, and read illegal data.

Conditions under which this bug occurs:

This bug may occur when all the following conditions are satisfied.

- CS spaces for which the speculative read function is valid/not valid exist, and each CS space is accessed.
- An undivided internal system clock is used for the external BUSCLK.
- The wait settings for the CS space for which the speculative read function is not valid satisfy the following two conditions.
  - The number of idle states is set to a value other than 0 (using the BCC register).
  - The number of data waits (set by the DWC0 and DWC1 registers) and address setup waits (set by the ASC register) are both set to 0.

[Workaround]

Implement any of the following workarounds to avoid this bug.

- Set the number of waits (using the DWC0 and DWC1 registers) or address setup waits (using the ASC register) for all the CS spaces to which an SRAM or page-ROM interface device is connected to 1 or more.
- Do not use the speculative read function for any CS space.

No. 11 Restriction on A/D converter

[Description]

The A/D converter may not be started in timer trigger mode.

[Workaround]

The function equivalent to timer trigger mode can be implemented using either of the following workarounds.

- Set the ADCE bit of the ADM0 register to 1 in the timer processing routine and then start the A/D converter in A/D trigger mode. (Startup of the A/D converter is delayed if a timer interrupt is held pending. Therefore, this method requires sufficient system verification.)
- Input a timer output signal (TOCn or TO1m) of timer C or timer ENC1 to the ADTRG pin for 500 ns or longer and then start the A/D converter in external trigger mode (n = 0 to 5, m = 0 or 1).